Lecture 4 Date, Time Objects, Return and Plot

Ziwen Ye

Stevens Institute of Technology

zye2@stevens.edu

September 22, 2018

Outline

Why we need to learn Date class

The core data object for holding data in R is the *data.frame object.* A date.frame is a rectangular data object whose columns can be of different types (e.g., numeric, character, logical, Date, etc.). The data.frame object, however, is not designed to work efficiently with time series data. In particular, subsetting and merging data based on a time index is cumbersome and transforming and aggregating data based on a time index is not at all straightforward. Furthermore, the default plotting methods in R are not designed for handling time series data. Hence, there is a need for a flexible time series class in R with a rich set of methods for manipulating and plotting time series data.

as.Date

as.Date is a functions to convert between character representations and objects of class "Date" representing calendar dates.

Example

```
> myDate <- as.Date("1970-01-01")
> mode(myDate)
[1] "numeric"
> # [1] "numeric"
> class(myDate)
[1] "Date"
> # [1] "Date"
> # internally it is number of days since Jan 01, 1970
> as.numeric(myDate)
[1] 0
> # [1] 0
```

as.Date

In *as.Date* it only support 1 single object rather than 2

Example

```
> #Bad example
> myDates <- as.Date("2013-01-01", "2013-01-02")
> myDates
[1] NA
> # doesn't work with 2 variables, need to pass a vector
>
> #Good example
> myDates <- as.Date(c("2013-01-01", "2013-01-02"))
> as.numeric(myDates)
[1] 15706 15707
```

as.Date

In *as.Date* it can only recognize some standard time type, other type of time it may not able to recognize.

Example

```
as.Date("1970/01/01")
# [1] "1970-01-01"
as.Date("1970-01-01")
# [1] "1970-01-01"
as.Date("01/01/1970") #non-standard input
# [1] "0001-01-19"

# set up time format when generate time data
as.Date("1970-01-01", format = "%Y-%m-%d")
# [1] "1970-01-01"
as.Date("02/24/2015", format = "%m/%d/%Y")
# [1] "2015-02-24"
```

Format codes for dates

Code	Value	Example
%d	Day of the month	23
%m	Month	01
%b	Month(abbreviated)	Jan
%B	Month(full)	January
%y	Year(2 digits)	90
%Y	Year(4 digits)	1990

When change date type, you can use *format()*.

Other useful function

- months()
- quarters()
- weekdays()
- seq.Date()
- seq()

Introduction

There are two POSIXt subclasses available in R: POSIXct and POSIXlt. The POSIXct class represents datetime values as the signed number of seconds (which includes fractional seconds) since midnight GMT (UTC universal time, coordinated) 19700101. This is analogous to the Date class with addition of times during the day. The POSIXlt class represents datetime values as a named list with elements for the second (sec), minute (min), hour (hour), day of the month (mday), month (mon), year (year), day of the week (wday), day of the year (yday), and daylight savings time flag (isdst), respectively.

Main difference from POSIXct and POSIXlt

Example

```
myDateTime1 <- as.POSIXct("01,01,2014 10:20:20",
                          format = "%d,%m,%Y %H:%M:%S")

myDateTime2 <- as.POSIXlt("01,01,2014 10:20:20",
                          format = "%d,%m,%Y %H:%M:%S")

myDateTime1$sec
# Error: object 'myDatetime1' not found
myDateTime2$sec
# [1] 20
```

strptime() and difftime()

Example

```
> # strptime()
> dt1 <- strptime("2013.12/23 01:00:34", "%Y.%m/%d %H:%M:%S")
> dt2 <- strptime("2013.12/23 01:02:37", "%Y.%m/%d %H:%M:%S")
> dt1
[1] "2013-12-23 01:00:34 EST"
> dt2$sec
[1] 37
>
> #You get time difference directly by using "-"
> dt2-dt1
Time difference of 2.05 mins
> #use difftime can change unit
> difftime(dt2, dt1, units = "secs")
Time difference of 123 secs
```

Brief summary

- strptime() is the most basic function, it only accept character as input. This is why this function is much faster in dealing with large input.
- strptime() and as.POSIXlt() returns a list
- as.POSIXct() returns a string
- Be extra careful when you set up the time zone

Return

Why use return series instead of price series.

- Return is a complete and scale-free summary of an asset.
- Return has more attractive statistical properties. (weakly stationary)

Return

Simple Return

$$1 + R_t = \frac{P_t}{P_{t-1}}$$

$$\therefore R_t = \frac{P_t}{P_{t-1}} - 1 = \frac{P_t - P_{t-1}}{P_{t-1}}$$

Log Return (countinuously compounded return)

$$r_t = log(1 + R_t) = log(P_t/P_{t-1})$$

$$= log(P_t) - log(P_{t-1})$$

Return

For multiple time intervals, say K:

Simple

$$1 + R_{t[k]} = \frac{P_t}{P_{t-k}} = \frac{P_t}{P_{t-1}} \frac{P_{t-1}}{P_{t-2}} \cdots \frac{P_{t-k+1}}{P_{t-k}}$$

$$= (1 + R_t)(1 + R_{t-1})(1 + R_{t-2})$$

Log

$$r_{t[k]} = log(1 + R_{t[k]}) = log[\frac{P_t}{P_{t-1}} \frac{P_{t-1}}{P_{t-2}} \cdots \frac{P_{t-k+1}}{P_{t-k}}]$$

$$= log(\frac{P_t}{P_{t-1}}) + \cdots + log(\frac{P_{t-k+1}}{P_{t-k}}) = \sum_{j=1}^{k-1} r_j$$

Continuously Compounding

one year: $\lim_{m\to\infty}(1+\frac{r}{m})^m = e^r$

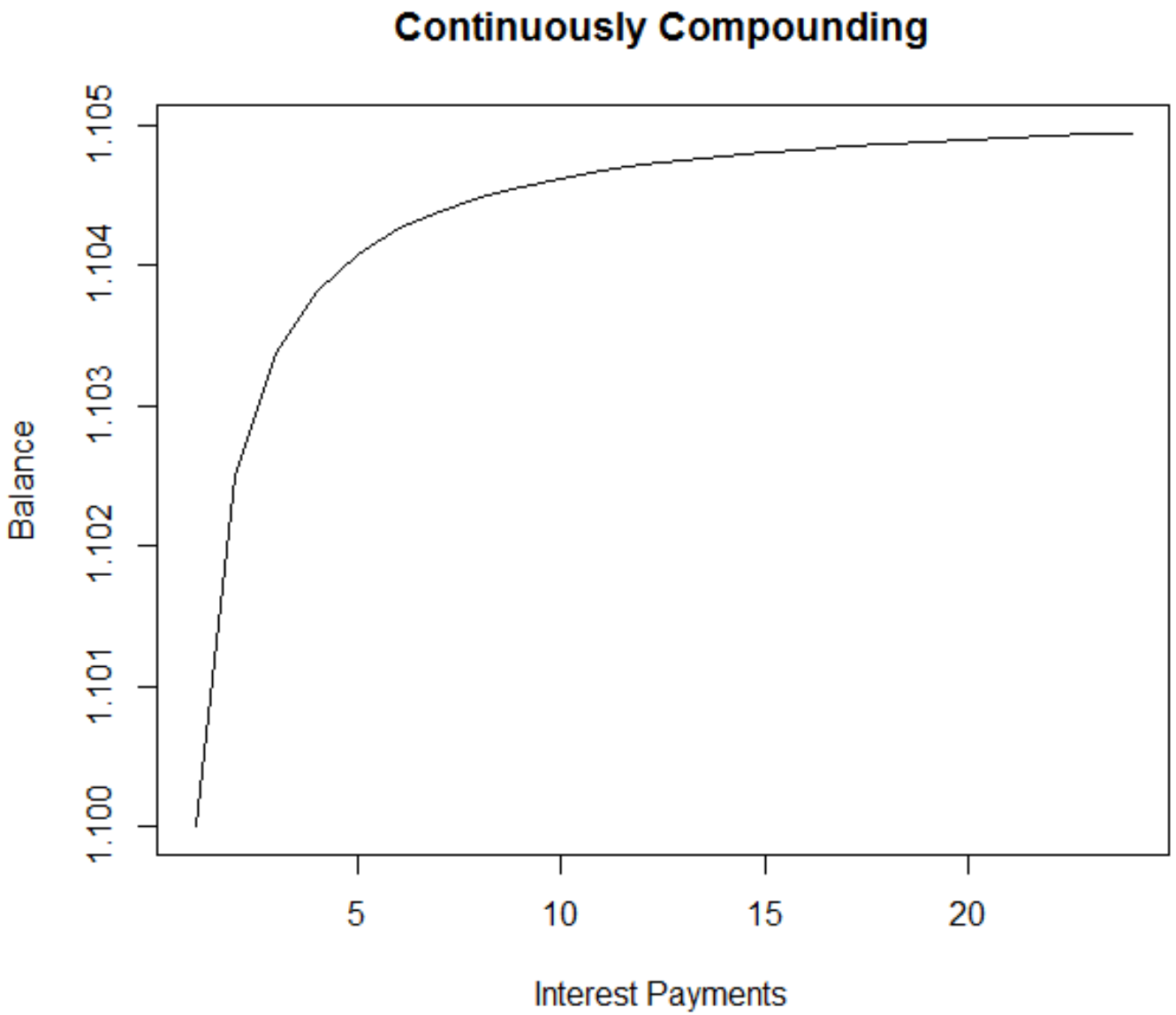
t years: $\lim_{m\to\infty}((1+\frac{r}{m})^m)^t = e^{rt}$

Continuously Compounding

Example

```
> result <- NULL
> x <- 1:24
> for (m in x)
+ {
+  result <- c(result, contConpound(0.1, m))
+ }
> plot.ts(result, main = "Continuously Compounding",
          xlab = "Interest Payments", ylab = "Balance")
```

Continuously Compounding



plot()

plot() is the most basic graphic function in R, it doesn't need any package to start. This function is able to generate line scatter plot, line graph, bar graph, histogram and box plot.

plot()

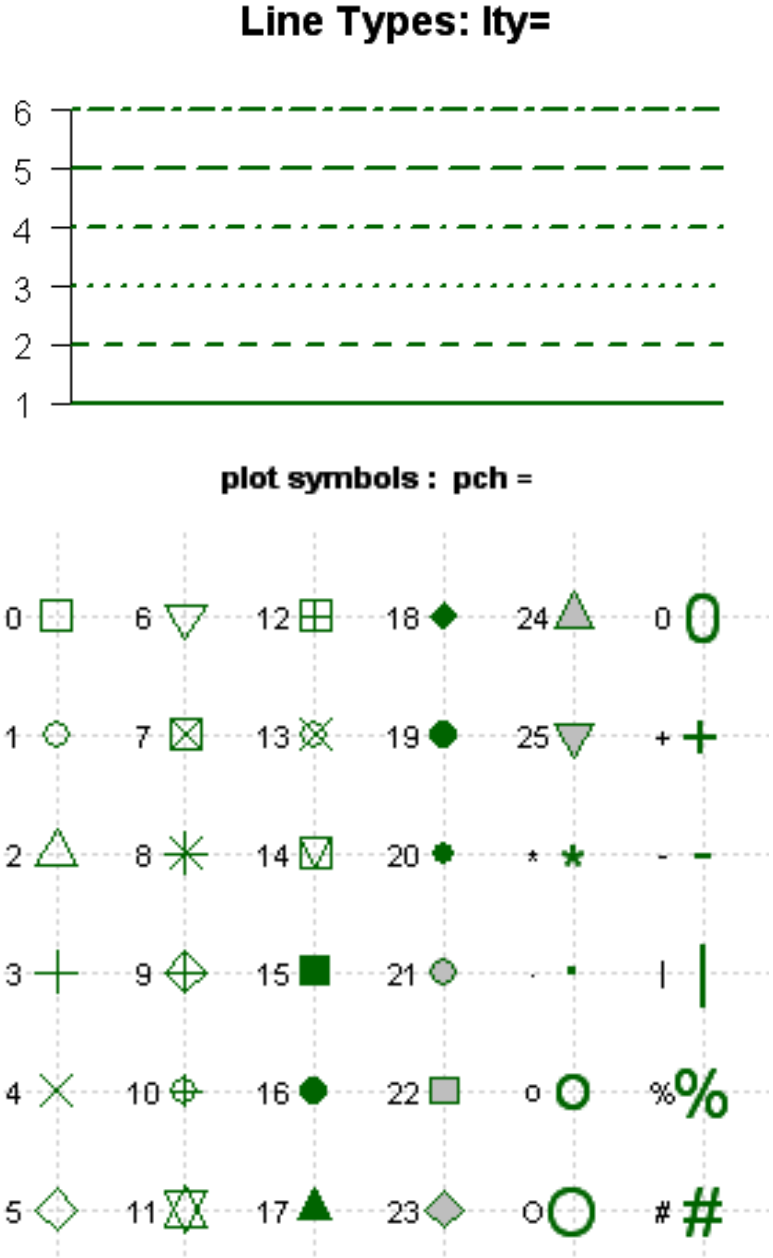
plot(x, y, ...)

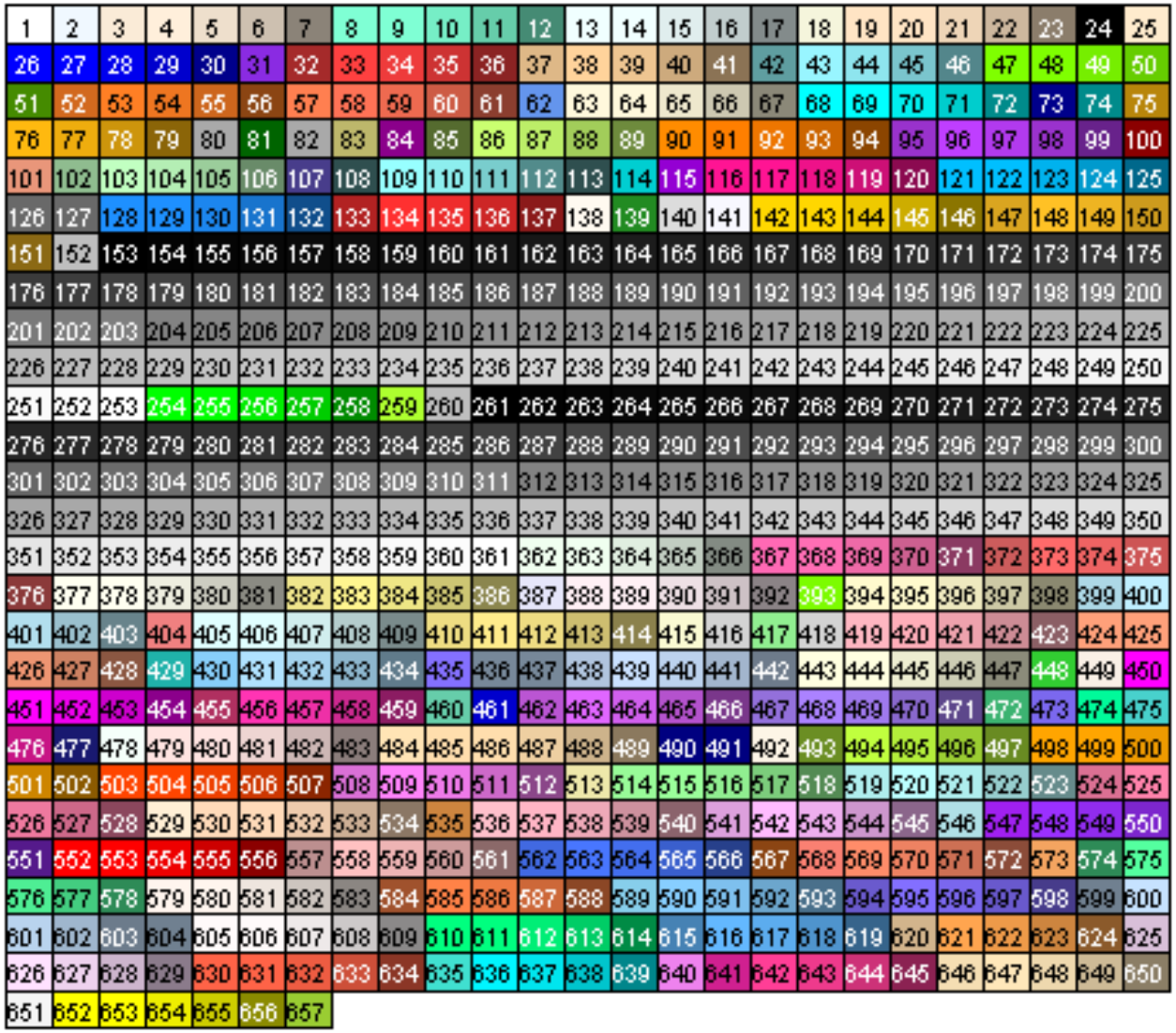
x means inputs in x-axis
y means inputs in y-axis

Advanced arguments

In plot() function, user is able to adjust plot format with following arguments.

Argument	Description
main	title for the plot
xlab,ylab	label for x or y axis
xlim,ylim	value range on x or y axis
lwd	line width
lty	type of line
pch	type of point
col	color for line or point


Line Types: lty=
6
5
4
3
2
1
plot symbols : pch =
0
6
12
18
24
0
1
7
13
19
25
+
2
8
14
20
*
-
3
9
15
21
.
|
4
10
16
22
o
%
5
11
17
23
O
#



Also you can refer to this
http://www.stat.columbia.edu/ tzheng/files/Rcolor.pdf

Scatter plot

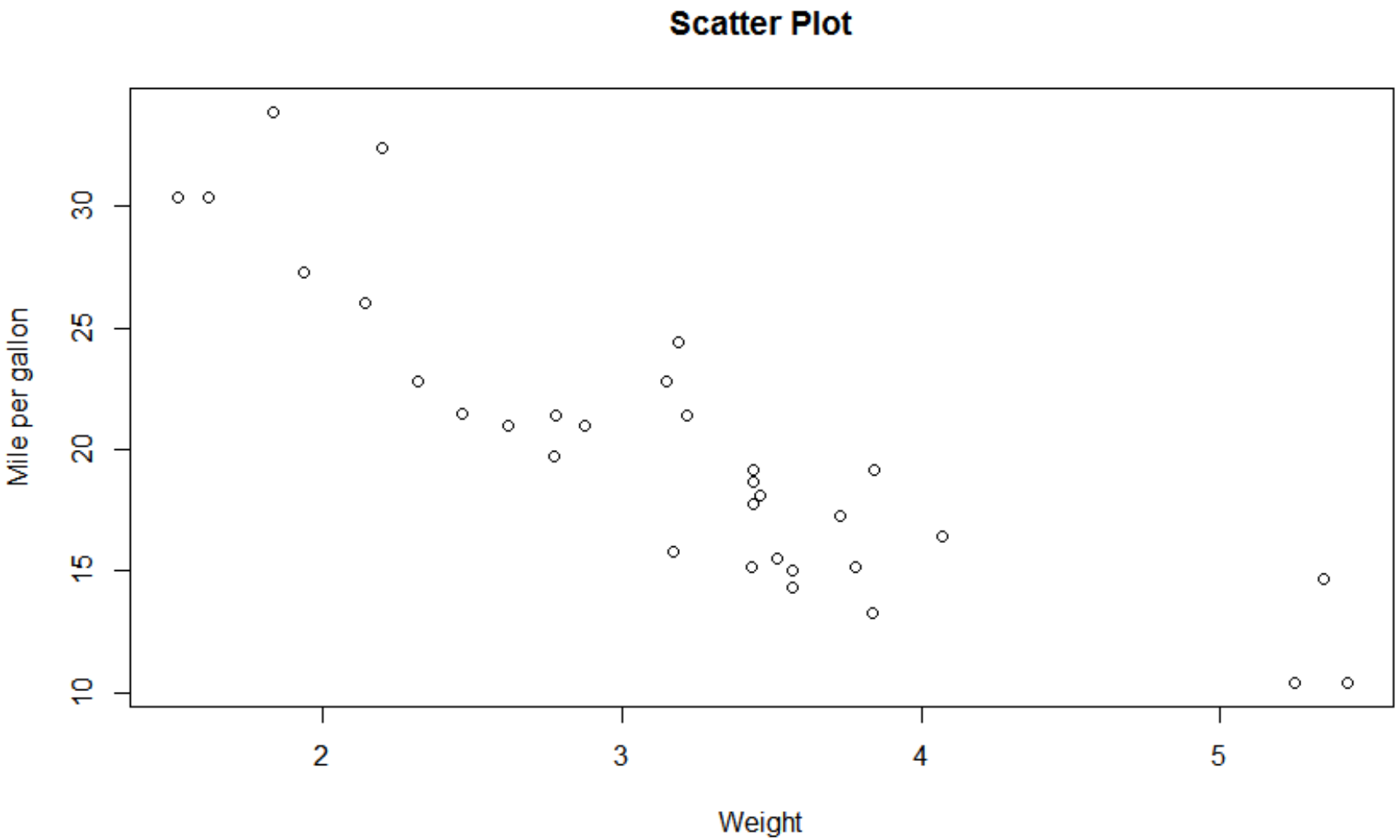
Scatter plot is a plot use coordinate to describe relationship between two variable

Example

```
plot(mtcars$wt, mtcars$mpg)
```

Advanced setting

1. Adding labels and title
2. Change point type


Scatter Plot
Mile per gallon
10
15
20
25
30
2
3
4
5
Weight

Bar plot

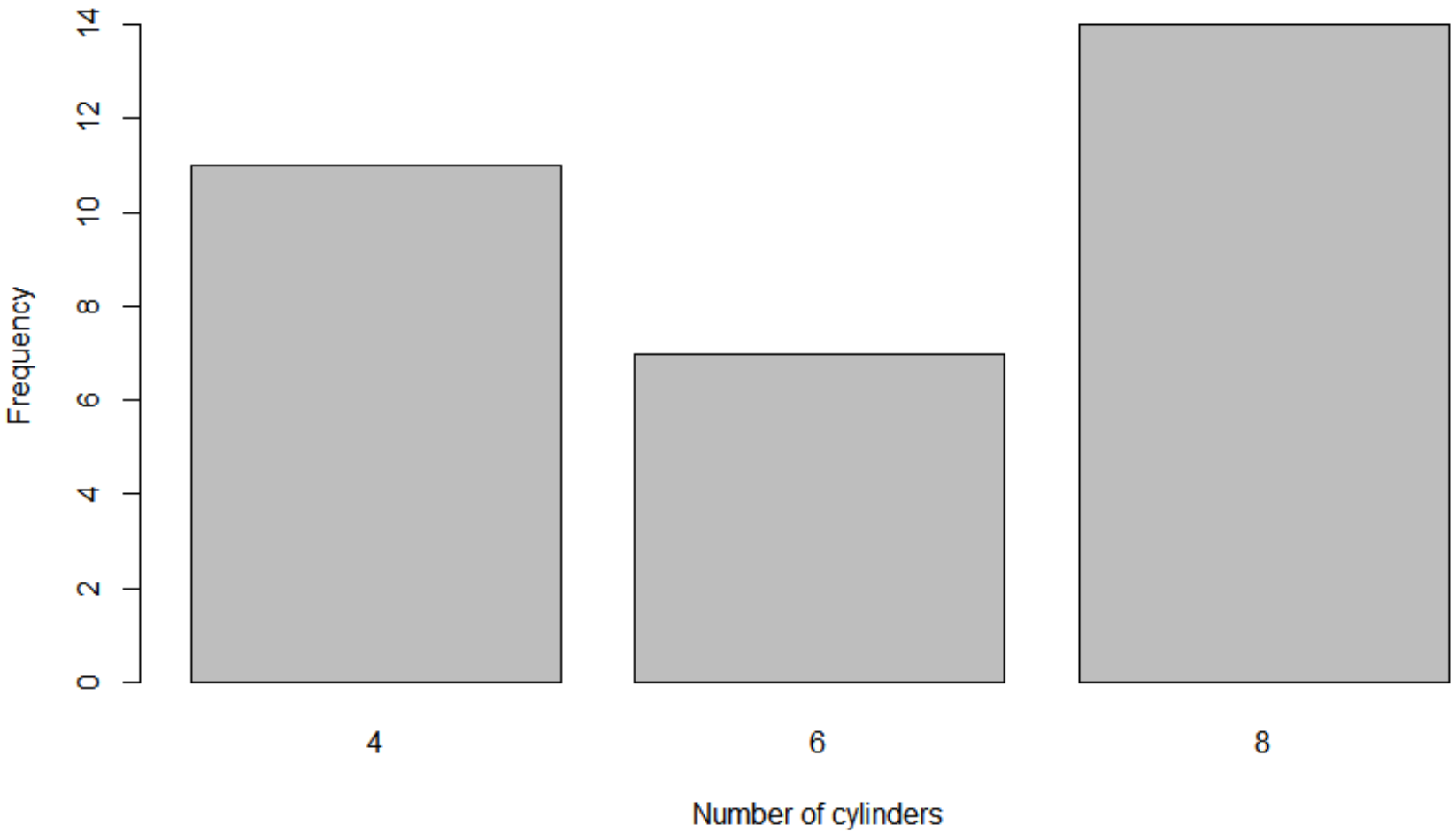
Bar plot can represent grouped data with rectangular bars

Example

```
barplot(table(mtcars$cyl))
```

Advanced setting

1. Change color of bar
2. Create stacked bar plot


Frequency
0
2
4
6
8
10
12
14
4
6
8
Number of cylinders

Legend

Legend is an important part for any plot, this is the place to identify your plot content.

Example

legend("location", names, fill=cols)

In most case, location, names and cols are the most basic information to put in legend.

For detailed information please check code.

Box plot

In descriptive statistics, a box plot is a convenient way of graphically depicting groups of numerical data through their quartiles.

Example

boxplot(len $\sim$ supp + dose, data = ToothGrowth)

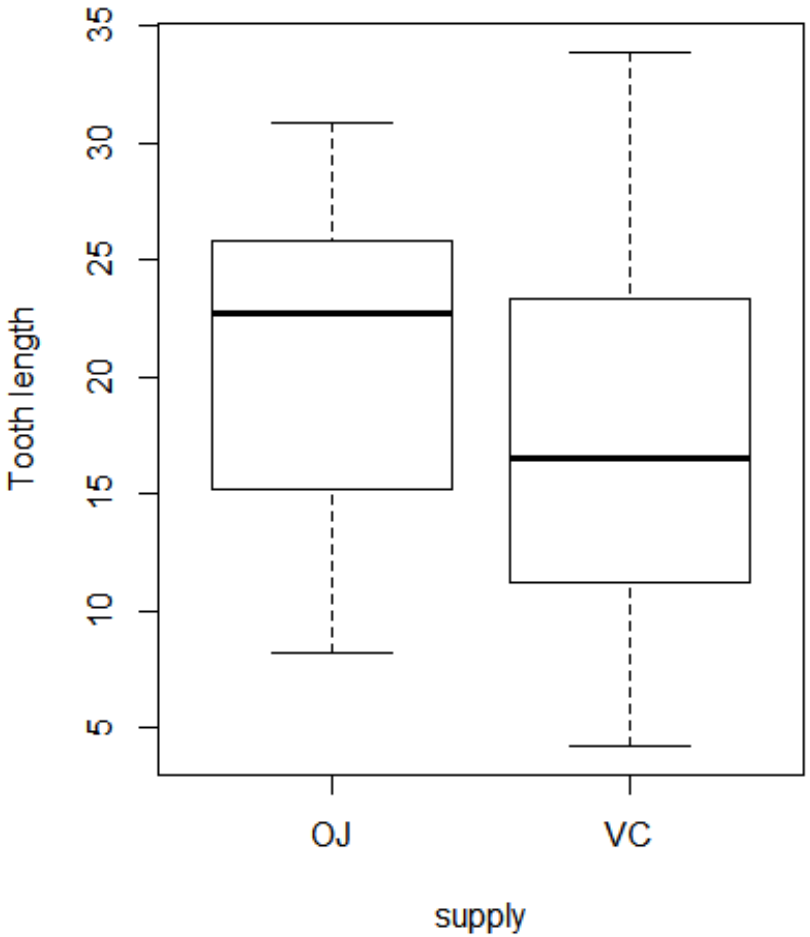
Advanced setting

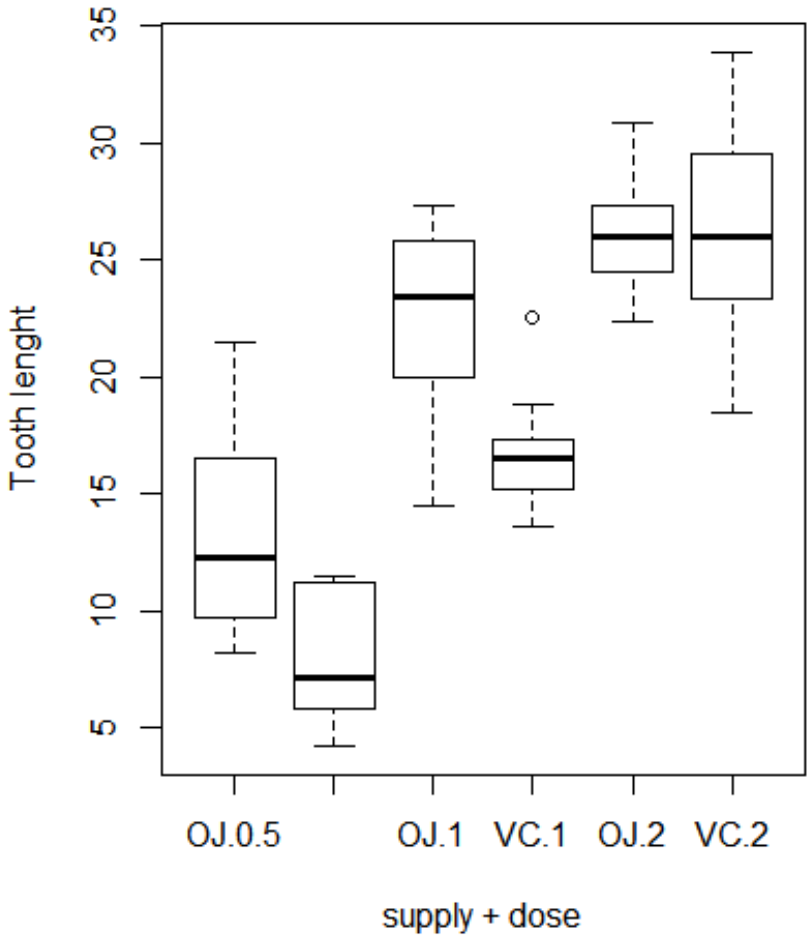
1. color for border and bar
2. point style for outlier
3. change median number color

Plots layout

If you want to re-arrange the location of the plot, you need to refer par() or layout(). No matter which one you are using, it will relocate the plot by "matrix format". However, layout() is more flexible.

- par() example is shown in the box plot section
- layout() example is shown in the histogram section


Tooth length
5
10
15
20
25
30
35
OJ
VC
supply


Tooth lenght
5
10
15
20
25
30
35
OJ.0.5
OJ.1
VC.1
OJ.2
VC.2
supply + dose

Box plot summary

- The format of box plot is boxplot(Y value X value)
- The input of Y-value can only be **numeric** vector
- The input of X-value do not have limitation (in most case, it is a factor vector)

Rewrite description for axes

To rewrite description for axes, we need to delete original content before we generate plot, or new description will overlap with old content.

Delete description

xaxt = "n"
yaxt = "n"

Above function can not be used individually, they are arguments in plot() function.

Change description for line chart

Sometime the line chart can not present a ideal output for time objects. In such case, removing the description from corresponding axis is the best solution.

For detailed information please refer to code.

Key word

axis()
axis.POSIXct()